Proposing a New Foundation of Attack Trees in Monoidal Categories

Harley Eades III

Computer and Information Sciences, Augusta University, Augusta, GA,
heades@augusta.edu

Abstract. TODO

1 Introduction

What do propositional logic, multisets, directed acyclic graphs, source sink graphs, Petri nets, and Markov processes all have in common? They are all mathematical models of attack trees [**?**], but even more than that, they can all be modeled in some form of a symmetric monoidal category[1] [**?**] – for the definition of a symmetric monoidal category see Appendix A. Taking things a little bit further, monoidal categories have a tight correspondence with linear logic through the beautiful Curry-Howard-Lambek correspondence [**?**]. This correspondence states that objects of a monoidal category correspond to the formulas of linear logic and the morphisms correspond to proofs of valid sequents of the logic. I propose that attack trees – in many different flavors – be modeled as objects in monoidal categories, and hence, as formulas of linear logic.

The Curry-Howard-Lambek correspondence is a three way relationship:

Categories	$\iff$	Logic	$\iff$	Functional Programming
Objects	$\iff$	Formulas	$\iff$	Types
Morphisms	$\iff$	Proofs	$\iff$	Programs

By modeling attack trees in monoidal categories we obtain a sound mathematical model, a logic for reasoning about attack trees, and the means of constructing a functional programming language for defining attack trees (as types), and constructing semantically valid transformations (as programs) of attack trees.

Linear logic was first proposed by Girard [**?**] and was quickly realized to be a theory of resources. In linear logic, every hypothesis must be used exactly once. Thus, formulas like $A \otimes A$ and A are not logically equivalent – here $\otimes$ is linear conjunction. This resource perspective of linear logic has been very fruitful in computer science. It has lead to linear logic as being a logical foundation of concurrency [**?**] where formulas may be considered as processes. This perspective fits modeling attack trees perfectly, because they essentially correspond to concurrent processes.

[1] We provide a proof that the category of source sink graphs is monoidal in Appendix B.

Girard's genius behind linear logic was that he isolated the structural rules – weakening and contraction – by treating them as an effect and putting them inside a comonad called the of-course exponential denoted $!A$. In fact, $!A \otimes !A$ is logically equivalent to $!A$, and thus, by staying in the comonad we become propositional. This implies that a modal of attack trees in linear logic also provides a model of attack trees in propositional logic, and a combination of the two. It is possible to have the best of both worlds.

In this short paper I introduce a newly funded research project[2] investigating founding attack trees in monoidal categories, and through the Curry-Howard-Lambek correspondence deriving a new domain-specific functional programming language called Lina for Linear Threat Analysis. We begin by defining the style of attack trees we will study here in Section 2, then we give a semantics of attack trees in a model of full intuitionistic linear logic in Section 3, we discuss how the semantics may be further abstracted in Section 4, and finally, we discuss the design of Lina in Section 5.

2 Attack Trees

In this paper I only consider basic attack trees with sequential composition which are due to Jhawar et al. [1], but one of our ultimate goals is to extend attack trees with more operators driven by are choice of semantics. The syntax for attack trees is defined in the following definition.

Definition 1. *The following defines the syntax of* ***Attack Trees*** *given a set of base attacks* $b \in \mathsf{B}$*:*

$$t ::= b \mid t_1 + t_2 \mid t_1 \sqcup t_2 \mid t_1 ; t_2$$

We denote parallel composition by $t_1 + t_2$*, choice between attacks* t_1 *and* t_2 *by* $t_1 \sqcup t_2$*, and finally sequential composition of attacks by* $t_1 ; t_2$*.*

The syntax given in the previous definition differs from the syntax used by Jhawar et al. [1]. First, I use infix binary operations, while they use prefix n-ary operations. However, we do not sacrifice any expressivity, because as we will see in the next section each operation is associative, and parallel composition and choice are symmetric. Thus, we can embed Jhawar et al.'s definition of attack trees into the ones defined here. The hard part of this embedding is realizing that the n-ary version of sequential composition can be modeled by the binary version, but if we have $\mathsf{SAND}(t_1, t_2, \ldots, t_n)$, then it is understood that t_1 is executed, then t_2, and so on until t_n is executed, but this is exactly the same as $t_1 ; t_2 ; \cdots ; t_n$.

The second major difference is that we denote parallel composition with an operator that implies we can think of it as a disjunction, but Jhawar et al. and others in the literature seem to use an operator that implies that we can

[2] This material is based upon work supported by the National Science Foundation CRII CISE Research Initiation grant, "CRII:SHF: A New Foundation for Attack Trees Based on Monoidal Categories", under Grant No. 1565557.

think of it as a conjunction. The semantics, however, tells us that it is really a disjunction. The parallel operation defined on source sink graphs defined by Jhawar et al. [1] can be proven to be a coproduct – see Appendix B – and coproducts categorically model disjunctions. Furthermore, parallel composition is modeled by multiset union in the multiset semantics, but we can model this has a coproduct. Lastly, the semantics we give in the section models parallel composition as a disjunctive operation. Thus, I claim that an operator that reflects this is for the better.

The third, and final difference is that we denote the choice between executing attack t_1 or attack t_2, but not both by $t_1 \sqcup t_2$ instead of using a symbol that implies that it is a disjunction. This fits very nicely with the semantics of Jhawar et al., where they collect the attacks that can be executed into a set. The semantics we given in the next section models choice directly.

3 Concrete Semantics of Attack Trees in Dialectica Spaces

4 Abstract Semantics of Attack Trees in Monoidal Categories

5 Lina: A Domain Specific PL for Threat Analysis

6 Conclusion and Future Work

References

1. Ravi Jhawar, Barbara Kordy, Sjouke Mauw, SaÅ!'a RadomiroviÄ, and Rolando Trujillo-Rasua. Attack trees with sequential conjunction. In Hannes Federrath and Dieter Gollmann, editors, *ICT Systems Security and Privacy Protection*, volume 455 of *IFIP Advances in Information and Communication Technology*, pages 339–353. Springer International Publishing, 2015.

A Symmetric Monoidal Categories

This appendix provides the definitions of both categories in general, and, in particular, symmetric monoidal closed categories. We begin with the definition of a category:

Definition 2. *A **category**, $\mathcal{C}$, consists of the following data:*

- *A set of objects $\mathcal{C}_0$, each denoted by A, B, C, etc.*
- *A set of morphisms $\mathcal{C}_1$, each denoted by f, g, h, etc.*
- *Two functions src, the source of a morphism, and tar, the target of a morphism, from morphisms to objects. If $\mathsf{src}(f) = A$ and $\mathsf{tar}(f) = B$, then we write $f : A \longrightarrow B$.*

- *Given two morphisms* $f : A \longrightarrow B$ *and* $g : B \longrightarrow C$*, then the morphism* $f; g : A \longrightarrow C$*, called the composition of* f *and* g*, must exist.*
- *For every object* $A \in \mathcal{C}_0$*, the there must exist a morphism* $\mathsf{id}_A : A \longrightarrow A$ *called the identity morphism on* A*.*
- *The following axioms must hold:*
  - *(Identities) For any* $f : A \longrightarrow B$*,* $f; \mathsf{id}_B = f = \mathsf{id}_A; f$*.*
  - *(Associativity) For any* $f : A \longrightarrow B$*,* $g : B \longrightarrow C$*, and* $h : C \longrightarrow D$*,* $(f; g); h = f; (g; h)$*.*

Categories are by definition very abstract, and it is due to this that makes them so applicable. The usual example of a category is the category whose objects are all sets, and whose morphisms are set-theoretic functions. Clearly, composition and identities exist, and satisfy the axioms of a category. A second example is preordered sets, $(A, \leq)$, where the objects are elements of A and a morphism $f : a \longrightarrow b$ for elements $a, b \in A$ exists iff $a \leq b$. Reflexivity yields identities, and transitivity yields composition. See the usual introductions for more examples [**?**].

Symmetric monoidal categories pair categories with a commutative monoid like structure called the tensor product. They are a categorical semantics of linear logic [**?**].

Definition 3. *A* ***symmetric monoidal category (SMC)*** *is a category,* $\mathcal{M}$*, with the following data:*

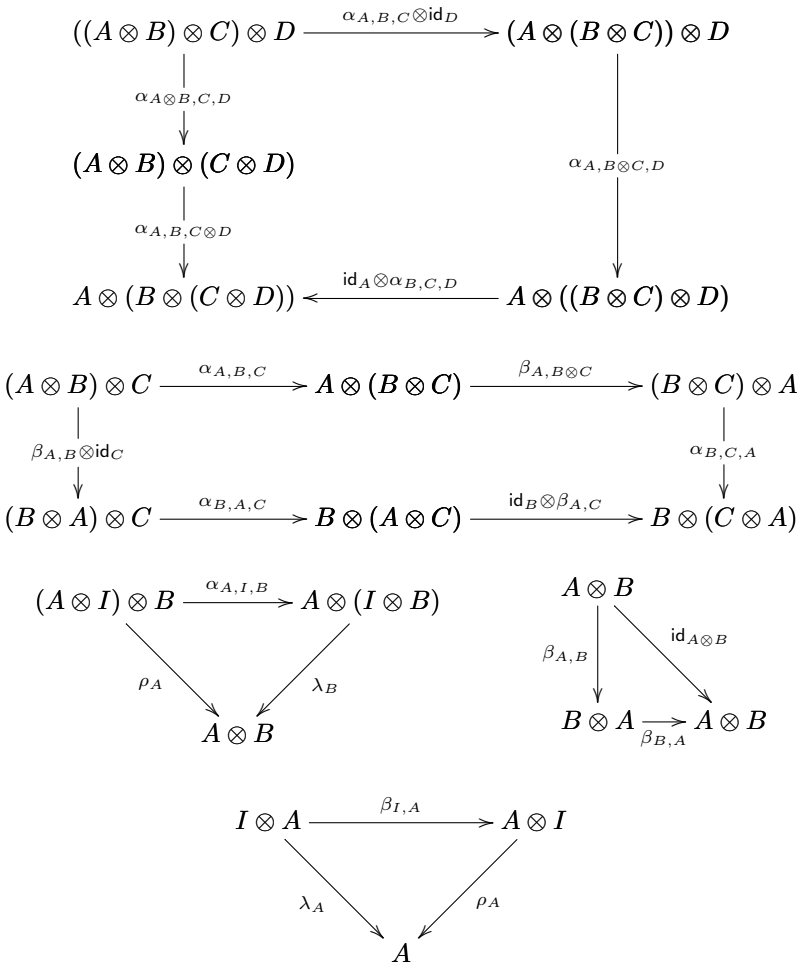
- *An object* I *of* $\mathcal{M}$*,*
- *A bi-functor* $\otimes : \mathcal{M} \times \mathcal{M} \longrightarrow \mathcal{M}$*,*
- *The following natural isomorphisms:*

$$\lambda_A : I \otimes A \longrightarrow A$$
$$\rho_A : A \otimes I \longrightarrow A$$
$$\alpha_{A,B,C} : (A \otimes B) \otimes C \longrightarrow A \otimes (B \otimes C)$$

- *A symmetry natural transformation:*

$$\beta_{A,B} : A \otimes B \longrightarrow B \otimes A$$

– *Subject to the following coherence diagrams:*



B Source Sink Graphs are Symmetric Monoidal